Süsteemprogrammeerimine keeles C

Loeng 2

milles jätkame sisendi ja väljundiga tutvumist, räägime andmetüüpide teisendamisest, tähemärgi andmetüübist, massiividest ja pointeritest

Eelmises osas (C keelena)

- C keel
  - primitiivne, aga kiire
  - eeldab programmeerijalt distsipliini
  - ei halda mälu
  - annab selgema tunnetuse arvutis toimuvast

Eelmises osas (Muutujad)

- Muutuja
  - mälus olev koht, millel on nimi
  - enne kasutamist deklareerida tüüp
  - deklaratsioonid koodibloki alguses
- Primitiivsed andmetüübid
  - string ei ole primitiivne andmetüüp
  - `sizeof()` operaator ütleb suuruse baitides

```
{
int i;
float f;

...

}
```

Eelmises osas (Integer)

- *Integer* jaguneb märgiga ja märgita täisarvudeks

```
int i /* signed */
unsigned int i /* unsigned */
```

- Täisarvu baitide järjekord mälus sõltub arhitektuurist (*byte order*)

Eelmises osas (programmikomponendid)

- Expression (tehe)
  - väärtustub
  - operator, operand

```
c = 'T'          >>>> 'T'
C + 6            >>>> 'Z'
9 > limit        >>>> '0'
```

- Statement
  - ei väärtustu (või teeb seda kõrvalefektina)

```
i = 6;
if ( i > 5 ) c = 6;
printf("abrakadabra\n");
```

Eelmises osas (tsüklid)

```
for (i = 0; i < 5; i++)
    statement


while (i < 5)
    statement


do
    statement
while (test);    /* ; on oluline meelde jätta */
```

Eelmises osas (funktsioonid)

- Funktsiooni defineerimine:

```
funktsiooni tüüp funktsiooni nimi(argumendid) { sisu }
int square(int x) { return x * x; }
```

- Funktsiooni prototüüp
  - näitab kompilaatorile funktsiooni argumentide arvu ja tüübi ja tagastatava väärtuse tüübi
  - funktsiooni kasutamiseks peab kompilaatorile näitama funktsiooni definitsiooni või prototüüpi

```
int double(int x);
```

Praktikumi moraal

- C programmi kompileerimine koosneb neljast etapist
  - *preprocessing* (# -ga algavad read)
  - *compilation proper* (tulemuseks programm assembleris)
  - *assembly (tulemuseks objektkood)*
  - *linking*
- Mõistlikud hoiatused argumendiga -Wall

```
gcc -Wall hello.c -o hello
```

Tänane kava

- Standarväljund ja sisend
- Tüübiteisendused
- char ja selle kasutamine programmis
- Massiivid ja pointerid

printf(string, param1, param2, ...)

- Esimene argument on kohustuslik string, edasised argumendid sõltuvad esimese sisust
- Formaadistringi sisuks on
  - tähemärgid
  - asendusmärgid, mis asendatakse vastavalt sisule ülejäänud argumentidega

printf() asendusmärgid

- Asendusmärkide kuju on järgmine: %[flag][width][.precision][length]*type*
  - flag: miinusmärk, joondab parameetri vasakule
  - **width** – minimalaalne asendusmärgi laius
  - **precision** – mitu kohta peale koma näidata
  - length – int puhul: h, kui short, l, kui long
  - *type* – määrab mis tüüpi muutuja aadressilt lugeda (täpsemalt järgmisel lehel)

Asendusmärkide tüübid

d, i – int; täisarvuna
o - int; kaheksandsüsteemis
x, X - int; 16ndsüsteemis abcdef või ABCDEF
u - int; *unsigned*
c - int; tähemärk
s - char *; tähed senikaua, kuni tuleb \0 märk
f - double; ujukomaarv (vaikimisi 6 kohta)
e - double; ujukomaarv eksponentidena
% - argumente ei vaadata, saad %-märgi

Printf() näited

```
int x = 10; double a = 6.0; double b = 5.8888;

printf ("eriti igav näide\n");
```

```
eriti igav näide
```

Printf() näited

```
int x = 10; double a = 6.0; double b = 5.8888;

printf ("%d = %o oct = %X hex\n", x, x, x);
```

```
10 = 12 oct = A hex
```

Printf() näited

```
int x = 10; double a = 6.0; double b = 5.8888;

printf("x on %d, 50%% x-st on %d\n", x, x / 2);
```

```
x on 10, 50% x-st on 5
```

Printf() näited

```
int x = 10; double a = 6.0; double b = 5.8888;

printf ("a=%6.3f b=%6.3f\n", a, b );
```

```
a= 6.000 b= 5.889
   123456   123456
```

Üllatus! Konstandid

- Konstandid on viis defineerida korduvatele numbritele mõistlik nimi

  (NB! ei pea piirduma numbriga)
- Lahenduvad preprotsessoris

  `#define nimi asendustekst`

  NB! preprotsessori käskudel puudub semikoolon (miks?)

```
#define PI 3.14
#define ABSOLUTE_ZERO -273.15
#define TRUE 1
#define int float /* debug fun */
```

Konstantide näide

```
#include <stdio.h>
    #define LOWER 0          /* lower limit */
    #define UPPER 300        /* upper limit */
    #define STEP 20          /* step size */
main() {
    int fahr;

    fahr = LOWER;
    while(fahr <= UPPER) {
        printf ("%d\t%d\n", fahr, 5 * (fahr-32) / 9);
        fahr += STEP;
    }
}
```

Jätkame kavaga: Täisarvude jagamine

- Kui jagada täisarve, on jagatis samuti täisarv!

  5 / 2 = 2 (võetakse täisosa)

  3 / 2 = 1

  5 % 2 = 1 (täisarvu jagamisel tekkiv jääk)

- Näide:

```
int fahr;
    for (fahr = LOWER; fahr <= UPPER; fahr += STEP)
        printf ("%d\t%d\n", fahr, 5 * (fahr-32) /9);
```

```
0       -17
20      -6
40      4
60      15
```

Kuidas jagada nii, et tuleks koma?

Tüüpide teisendamine

- Teisendamine vägisi (*Explicit Casting*):
  (tüüp)muutuja_nimi

```
int thirteen = 13, seven = 6;
float f = (float)i / (float)j; /* f = 2,166666etc*/
```

Halva stiili näide: muutuja seven sisaldab väärtust 6

Teisendamine automaatselt (Implicit Casting)

- Kui liikmed on erinevatest tüüpidest, on tulemus mahukamat tüüpi

```
int i = 7; float f = 5.0;
float d = i / f       /* d = 1.4 */
```

- Konstandid erinevad: 5 on integer tüüpi, 5.0 float. Näiteks:

```
int i = 7;
double j = i/5;       /* j = 1.0 */
double k = i/5.0      /* j = 1.4 */
```

Korrektne Fahrenheiti programm

```
/* parandatud tsükli sisu */
int fahr;
for (fahr = LOWER; fahr <= UPPER; fahr += STEP)
    printf ("%3d %6.1f\n", fahr, (5.0/9.0)*(fahr-32));
```

```
  0   -17.8
 20    -6.7
 40     4.4
 60    15.6
```

Automaatsetest teisendustest veel

- Üldreegel teisendamisel: operaatorid teisendatakse kõige "võimekamaks"
- long double > double > float > long > int > short
- short ja char teisenduvad int-iks

- *signed* ja *unsigned* muudavad olukorda

Signed & Unsigned

- Kui operaatoritest üks on märgiga ja teine märgita, kuulub teisenduse tulemus ilma märgita andmetüüpi:

```
int i = -1; unsigned int u=0;
if (u > i)
    printf("Matemaatikud on rahul\n");
else
    printf("Matemaatikud vangutavad pead\n");
```

- i (väärtusega -1) teisendamisel märgita täisarvuks saame UINT_MAX = 4294967295
- Vältimine:

```
if ((long)u > (long)j)
```

Char tüüp

- Tähemärgikonstandid on ülakomade vahel
  - `'a' 'b' '1' '$'`
- Char on olemuselt number, seega on seda võimalik kasutada ka tehetes:

```
char c = 'a';
c++; /* = 'b' */
int k = (c < 'b'); /* =1 if c < 'b', else = 0 */
c = 65; /* numeric representation of A*/
```

- Numbriväärtused saab ASCII tabelist http://www.asciitable.com/ või `man ascii`

Erimärgid

- Mõned märgijadad on erilise tähendusega:

  \n reavahetus

  \b backspace

  \t tab

  \' '

  \" "

  \\ \

  \a kell või vile või piiks

  \x*kuueteistkümendarv* täht koodiga *kuueteistkümnendarv*
  näiteks \xa on täht koodiga 0xA

Tähemärkide sissesöömine ja väljutamine

- Standardlibra funktsioonid:

```
char c;
c = getchar (); /* võta standardsisendist tähemärk */
putchar (c); /*  topi tähemärk standardväljundisse */
```

- Kasutamiseks lisada <stdio.h>

Näide (vaese mehe cat)

- Ülesanne: Kopeeri sisend väljundiks

```
#include <stdio.h>

main() {
    int c; /* note int */
    while((c=getchar()) != EOF) {
        putchar(c);
    }
}
```

  - NB! EOF on defineeritud konstant, mille standardfunktsioonid sisendi lõppemisel tagastavad
  - Kirjutage programm, mis annab EOF-i numbriväärtuse

Kuidas lugeda sõnade arv failis?

- Algoritm:

```
state = OUT of word
number of words = 0
read next character
while the character is not EOF
  If character is whitespace
      state = OUT
  If character is not whitespace AND state=OUT
      state = IN
      increment number of words
  read next character
```

Nii lugeda sõnade arv failis!

- Programm

```
#include <stdio.h>
#define IN 0
#define OUT 1
main() {
   int c, nw, state;
   state = OUT;
   nw = 0;
   while ((c = getchar()) != EOF) {
       if (c == ' ' || c == '\n' || c == '\t')
              state = OUT;
       else if (state == OUT) {state = IN; ++nw; }
   }
   printf("there were %d words in the input\n", nw);
}
```

Nii oleks palju hirmsam ju!

- Programm

```
#include <stdio.h>
/* state 0 is in the word
   state 1 is out */
main() {
   int c, nw, state;
   state = 1;
   nw = 0;
   while ((c = getchar()) != EOF) {
       if (c == ' ' || c == '\n' || c == '\t')
              state = 1;
       else if (state == 1) {state = 0; ++nw; }
   }
   printf("there were %d words in the input\n", nw);
}
```

Loogikaoperaatorid

- Võrdlusavaldised:

  a<b, a>b, a<=b, a>=b, a==b, a!=b

  – Võrdsed 1-ga kui väärtus on tõene, 0 muidu. Tüübiks int

- Loogikaavaldised:

  !a NOT A

  a || b A OR B

  a && b A AND B

  NB! Ühekordne | ja & on bitthaaval tehted

Massiivid (*Array*)

- Deklareerimine

  tüüp nimi[suurus];

  ```
  int a[10]; /* massiiv 10 täisarvust */
  ```

- Elemendid: a[0], a[1], ..., a[9]
- a[10] ei ole massiivi element, vaid lihtsalt järgmine aadress mälus! Kompilaator seda ei kontrolli ja järgnev kompileerub ja paneb käivitamisel pange (juhul kui veab):

```
int a[10];
a[10] = 1; /* kirjutab massivi lõpust edasi */
```

Massiivide kasutamine

- Algväärtustamine:

```
int a[10] = {1, 2, 3, 4, 5, 6, 7, 8 , 9, 10};
```

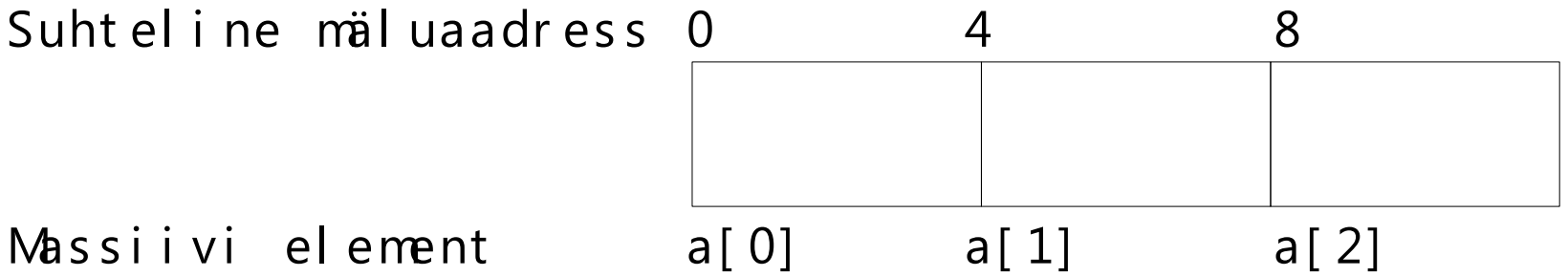
- Mitmemõõtmelised massiivid:

```
    /* 2 rida 3 tulpa */
    int a[2][3] = {{1, 1, 1}, {2, 2, 2}};
```

- Mitmemõõtmeliste juurde tuleme hiljem tagasi

Massiivid ja mälu

- Massiivi, mis defineeriti kui *int a[3]*, teine element (*a[1]*) paikneb suhtelisel aadressil *sizeof(int)*1*



- Adresseerimisel arvutab kompilaator massiivi elemendi asukoha järgmiselt:
  - aadress = baasaadress + index * sizeof(tüüp)

Pointerid

- Pointer – muutuja, mis hoiab mõne muu muutuja aadressi
- Defineerimine:

```
tüüp *nimi;  /* Siin on * oluline tähendus */
float f = 1.0;
float *fp;
fp = &f; /* fp sees on nüüd f aadress */
```

Pointerid

- "Address of" operaator **&**
  - *&muutuja* annab meile muutuja aadressi
- "Value of" operaator: ***
  - **pointer* annab meile pointeri poolt määratud mäluaadressi sisu

```
float f1 = 1.0, f2 = 2.0;
float *pf1;
pf1 = &f1; /* pf1 sees on f1 aadress */
f2 = *pf1; /* f2 = 1.0 */
```

Printf() näited

```
int x = 10;

printf ("x aadress on %#x\n", &x );
```

```
x aadress on ffffccb0
```

Otsa sai loenguaeg